A FUZZY LOGIC-BASED APPROACH FOR NODE LOCALIZATION IN MOBILE SENSOR NETWORKS

A Thesis

by

HARSHAVARDHAN CHENJI JAYANTH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2009

Major Subject: Computer Engineering

A FUZZY LOGIC-BASED APPROACH FOR NODE LOCALIZATION IN MOBILE SENSOR NETWORKS

A Thesis

by

HARSHAVARDHAN CHENJI JAYANTH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Radu Stoleru
Committee Members,	Rabi Mahapatra
	Deepa Kundur
Head of Department,	Valerie Taylor

December 2009

Major Subject: Computer Engineering

ABSTRACT

A Fuzzy Logic-based Approach for
Node Localization in Mobile Sensor Networks. (December 2009)
Harshavardhan Chenji Jayanth, B.Tech, National Institute of Technology
Karnataka, Surathkal, India
Chair of Advisory Committee: Dr. Radu Stoleru

In most range-based localization methods, inferring distance from radio signal strength using mathematical modeling becomes increasingly unreliable and complicated in indoor and extreme environments, due to effects such as multipath propagation and signal interference. We propose FuzLoc, a range-based, anchor-based, fuzzy logic enabled system system for localization. Quantities like RSS and distance are transformed into linguistic variables such as *Low*, *Medium*, *High* etc. by binning. The location of the node is then solved for using a nonlinear system in the fuzzy domain itself, which outputs the location of the node as a pair of fuzzy numbers. An included destination prediction system activates when only one anchor is heard; it localizes the node to an area. It accomplishes this using the theoretical construct of virtual anchors, which are calculated when a single anchor is in the node's vicinity.

The fuzzy logic system is trained during deployment itself so that it learns to associate an RSS with a distance, and a set of distances to a probability vector. We implement the method in a simulator and compare it against other methods like MCL, Centroid and Amorphous. Extensive evaluation is done based on a variety of metrics like anchor density, node density etc.

To My Parents,

Teachers That Were,

Teachers That Are

and

Teachers To Be

ACKNOWLEDGMENTS

I wish to thank my advisor Dr. Radu Stoleru, without whom this work would have been an unrealized dream. His methods of intuitive thinking, research methodology and outlook on life will be something I will treasure and learn from in the years to come. I would like to thank him for accepting me into the LENSS lab, which provided me with the necessary exposure to the field of sensor networks. I would also like to thank Dr. Deepa Kundur and Dr. Rabi Mahapatra for being part of my committee.

I also wish to thank all the members of the Laboratory for Embedded Networked Sensor Systems, especially Mike George, for their support and companionship during the course of my study. This work would not have been possible without their constructive criticism.

I wish to thank my parents CG and SJ for their constant love, support and encouragement. Last, but not the least, I would like to acknowledge my friends, especially Chandan, Raghu, Anil and Riddhi for making my stay in College Station memorable and fun-filled.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Localization is a service required by location-aware wireless sensor network (WSN) applications. Sometimes, the nodes in a network when deployed will need to know their position in terms of absolute or relative coordinates. Localization provides that service through a protocol. One way to do localization is by employing GPS devices on these nodes; however that is expensive both cost and energy-wise.

Ranging is a simple low-cost method based on received signal strength (RSS). Since the attenuation of a signal depends on distance between radios among other factors, ranging provides a fair tradeoff between accuracy and hardware requirement. In extreme environments however, it is difficult and sometimes impossible to expect a predictable relationship between distance and RSS. In extreme environments however, this method introduces large error due to multipath, fading etc. Fuzzy logic provides a way of "learning" about the environment so that distance can be correctly inferred from RSS. This is accomplished through a set of rules which is nothing but learned intelligence. The input is fed into the fuzzy inference system which consults the rules in order to compute an output. Note that the input and output can be any related quantity, not just RSS and distance. This basic technique has been employed in two constituent subsystems of FuzLoc - the Fuzzy Non Linear System (FNLS) and the Fuzzy Grid Prediction System (FGPS). FGPS uses fuzzy logic to relate a set of distances to a grid, and the probability that the node will be found in that particular

---

The journal model is *IEEE/ACM Transactions on Networking*.

grid.

The contributions to our thesis lie in the semantic treatment of input using fuzzy logic, which subsequently helps localize the node in the fuzzy domain itself and the underlying mathematical framework. The location will not be a geometric point, but rather a small area with the error encoded. We feel that localizing a node to a small area provides a good tradeoff against localizing to a point.

The rest of the thesis is organized as follows. In Chapter II we review and discusses related work. Chapter III introduces fuzzy logic. Chapter IV introduces the framework with ample discussion, descriptions and an example. Chapter V provides for an overview of the system, along with a distributed protocol for node localization. Chapter VI evaluates FuzLoc. Chapter VII discusses impact, implications and future work.

CHAPTER II

RELATED WORK

There has been a substantial amount of previous work dealing with the problem of localization in sensor networks. They can be classified into range-based/range-free and anchor-based/anchor-free methods independent of each other. Anchors are nodes which know their positions precisely. Ranging refers to the process of determining the distance/angle between 2 nodes from the radio characteristics between them. Range-free methods use only connectivity information, i.e, distance is inferred based on the contents of messages.

2.1 Range-free Localization Methods

These are also called coarse grained methods. Hop counting is a technique that is frequently used in these scenarios. The average hop length of the network is first computed and then the distance between two nodes will be inferred from the number of hops a packet takes. DV-hop [1] is one such work that uses this method. A major drawback is that hop-counting will fail for networks with irregular topologies such as those with a concave shape [2]. If the nodes are mobile, then this method incurs a lot of overhead since all the hop-counters will have to be refreshed every single time. Amorphous computing [3] also uses a similar approach.

Centroid [4] performs GPS-free indoor localization. The goal is achieved by simply taking the average of the co-ordinates of the anchors each node hears from. This method requires a large anchor density and fails when all the anchors are on the same side of the node. The advantage is, however, extremely low computational

resources are needed. The method introduces a large error for ad-hoc networks. APIT [5] is a similar method which divides the area of deployment into triangles formed by anchors and then estimates the location. It assumes a large anchor density and higher radio ranges for the anchor nodes.

Hu and Evans [6] have proposed a technique based on sequential Monte Carlo sampling. This technique has been widely used in robotics and hence handles mobility very well. The nodes statically estimate their positions using the positions of their one and two hop neighbors. The Monte Carlo method has been used in a similar fashion by the MSL [7] method. Both these methods consider radio range irregularity in extreme environments with the Degree of Irregularity (DoI) method. This essentially randomizes the radio range of a node in every direction so that nodes which are "near" cannot communicate because the radio range in that direction is different in time. Yet another similar method [8] uses special hardware to recover the mobile node's pedometry data and then uses Monte Carlo to localize.

2.2 Range-based Localization Methods

Fine grained methods refer to those that require an estimate of the distance or angle between two nodes to localize. A frequent requirement is the presence of at least three anchors so that basic uniqueness and geometric constraints are satisfied. The simplest method is, of course, GPS which uses the time of arrival of signals from satellites in order to obtain the precise location of a node in latitude-longitude format. A big drawback is increased size of nodes, high energy consumption and increased cost.

Some methods use special hardware for very accurate localization. Precise measurement of the phase difference between signals from 2 anchors is used to localize [9]. Expensive hardware is a major drawback. The spinning beacons localization

method [10] is an indigenous method that uses physically rotating beacons in order to take advantage of the Doppler Effect.

RADAR [11] is a method which uses surveying to predetermine RSSI values at any point in the area of deployment. Distance is then inferred from RSSI through this data. Time difference of arrival (TDoA) and angle of arrival (AoA) [12] are two fine grained methods requiring special hardware. Hybrid methods like parameter estimation have been used [13]. Work by Patwari et al. uses the Cramer-Rao bound to minimize the variance of error in location to iteratively localize [14].

Fuzzy logic has been proposed as a method to locate cellular phones in a hexagonal grid in a cellular network [15]. It assumes a fixed number of anchors but handles mobility very well. The computation and refining are not suitable for a resource-constrained computation platform like a micaZ node. This was the inspiration for this work.

Sensors equipped with optical sensors and reflectors [16] [17] have been used to localize accurately. These are very application specific; cst is another deterrent.

Some methods are anchor-free, i.e., they do not rely on the luxury of finding 2 or more anchors in their vicinity. Maps and map stitching and consequently graph embedding localize nodes based on inter-node distances. Such methods are computationally intensive. MDS-MAP [18] is one such method. Some others [19, 20] are worth mentioning. However, these methods require atleast 3 anchors to obtain absolute coordinates for the node in 2 dimensions.

2.3 Monte Carlo Based Localization Methods

The Monte Carlo method is widely used in the field of robot localization. The high processing power involved is not a deterrent as most robots are not as resource con-

strained as nodes. This method has been applied to sensor networks as well, with some adaptation for low processing and memory requirements.

MCL [6] uses the maximum velocity of the node to filter out incorrect samples. This way, mobility aids localization instead of hampering. It also uses knowledge of indirect seeds. However, it will be shown in this thesis that such sampling based methods perform badly when there is high noise/imprecision involved in sampling. There are many works that build upon MCL [21, 7, 22]. These works seek to reduce the processing power that MCL requires, and also reduces the error by constraining the sampling area [21]. MSL seeks to consider non-anchor neighbors for sampling, by considering already localized nodes. OTMCL is an innovative method which assumes that nodes know their direction, either by the use of extra hardware or otherwise. Errors in sensing the direction affect the accuracy, not to mention the energy cost required.

However, it should be noted that all these methods perform poorly in the face of extreme DoI. In other words, particle filtering does not handle imprecision well. For example, since these are anchor based methods, having a large number of anchors in a high DoI environment means that the amount of misinformation increases since these anchors contribute to the filtering process. This effect is shown in this thesis' evaluation section. Therefore, Monte Carlo based methods suffer from large memory requirement as well as the inability to handle imprecision.

CHAPTER III

PRELIMINARIES

We present our localization algorithm as a range based, anchor based one. The core intuition is that ranging can be accomplished by "learning" about the environment, and then using that intelligence to associate a RSS with a distance. Fuzzy logic (FL) provides a cheap, inexpensive way to use and store the learned intelligence. We feel that attempting to learn about the environment will eventually prove better than trying to model the environment using complicated means. This point is especially emphasized in extreme environments (like a typical indoor urban office with very many metallic objects) where phenomenon like multipath and ranging are not easy to model mathematically.

3.1 Definitions and Problem Formulation

A good method of localizing nodes in a highly mobile network is a distributed method, given that a centralized method requires significantly more processing time (ref map stitching). When each node individually localizes itself, we again have a choice between probabilistic estimation based methods (MCL, Cramer-Rao bound based MLE) and multilateration based methods. In this thesis, we treat localization as a problem of range-based multilateration (as opposed to localization from mere connectivity information). Both the ranging and the multilateration part occur in the fuzzy domain. We intend to represent the location as a fuzzy number, thus encoding the (in)accuracy information. In case the location is needed as a crisp number, an α-cut of the fuzzy number with a desired confidence threshold will return a crisp number.

Fuzzy Logic revisits classical set theory and enables it to have non-rigid (or fuzzy) set boundaries. For example, define a classical set `Far` to contain elements between and including 1000 and 2000. Then the number 999 would be excluded since it is a classical set. For a fuzzy set named Far, the number would still belong, albeit partially. This factor which denotes the associativity with the fuzzy set is called degree of membership, and is a scalar between 0 and 1. If variables like *distance* take on values such as 1000 or 42, linguistic variables like $DISTANCE$ take on non-numeric values like $Near$ or Far which represent imprecise information.

A "fuzzy bin" is synonym for a fuzzy set. For every fuzzy bin there is an associated membership function $\mu(x)$ which essentially defines the set. The purpose of the $\mu(x)$ function is to find any given crisp (non-fuzzy) number's degree of membership in that particular set. Note that a crisp number can belong to more than one fuzzy set(s) at a given time, with varying degrees of membership. In order to translate the crisp value into a fuzzy bin, we simply choose that bin in which it has the highest membership. A popular membership function is the triangular membership function defined as follows:

$$\mu(x) = \begin{cases} 0 & \text{if } x < a \\ (x-a)/(b-a) & \text{if } a \leq x \leq b \\ (c-x)/(c-b) & \text{if } b \leq x \leq c \\ 0 & \text{if } x > c \end{cases} \tag{3.1}$$

where (a, b, c) defines a triangular bin. For example, in Figure 1, the $SMALL$ bin can be represented as $(5, 10, 15)$ and $MEDIUM$ as $(10, 15, 20)$. A crisp number 13.75 has a membership of 0.25 in $SMALL$, whereas it has a membership of 0.75 in $MEDIUM$. The triangular membership function is a reasonable substitute for the Gaussian since it has linear components only, and not much computation is required

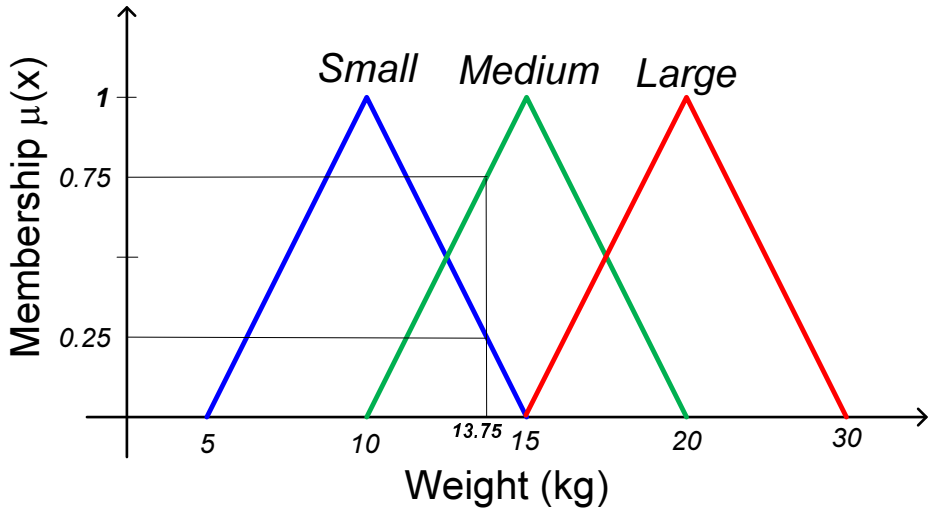

Fig. 1. Triangular membership functions

to compute the membership.

For a given fuzzy system, the fuzzy ruleset relates two linguistic variables in the form of an IF-THEN clause. Typically the IF clause contains the input linguistic variable (e.g., RSSI) and the THEN clause contains the output linguistic variable (e.g., DISTANCE). An example rule is:

IF [INPUT] is [*LOW*] **THEN** [OUTPUT] is [*MEDIUM-LARGE*]

A fuzzy number is a special fuzzy bin where the membership is 1 at one and only one point. A fuzzy number represents a multi-valued, imprecise quantity unlike a single valued traditional number. We propose to represent the two dimensional location of a node as a pair of fuzzy numbers (X, Y) where both X and Y are fuzzy numbers. The imprecision in a crisp number can be compacted into a single fuzzy number. This location is calculated in the fuzzy domain itself, beginning with the fuzzification of the signal strength received from anchor(s).

3.2 Assumptions

Anchor nodes are present. They are a fraction of total nodes. Anchor nodes know their locations. Anchor nodes have more sophisticated storage capabilities than regular nodes, to store the fuzzy rules.

Nodes do not know what their maximum velocity is. Nodes do not know the area of deployment beforehand. Nodes have a preset grid size - for eg, if the location is (250,300) nodes can intuitively assume a reasonable grid length of 100, and place itself in the grid (3,3). For latitude and longitudes in decimal format, for eg, they can assume a grid length of $1e-7$ degrees (roughly 10m). They can then assume there are 25 grids in total, and calculate the locations of those virtual anchors.

CHAPTER IV

FUZZY LOGIC BASED NODE LOCALIZATION

4.1 Fuzzy Multilateration

Consider a node about to be localized, as shown in Figure 2. It transmits a beacon message which is heard by three (in our example) anchors. Each anchor A_1, A_2 and A_3 measures the RSSI of this message. Using this RSSI, an anchor proceeds to determine the distance between it and the node. Suppose that the ruleset were as follows, where $RSSI_i$ and $Dist_i$ are fuzzy linguistic variables (e.g. LOW, $MEDIUM$, $HIGH$):

Rule 1: **IF** RSSI is $RSSI_1$ **THEN** DIST is $Dist_1$

Rule 2: **IF** RSSI is $RSSI_2$ **THEN** DIST is $Dist_2$

$\vdots$

Rule i: **IF** RSSI is $RSSI_i$ **THEN** DIST is $Dist_i$

We have the input RSSI from Anchor 1 as rss_1. The membership $\mu_i(rss_1)$ of this value is computed for each input bin $RSSI_i$, and that value is to the corresponding output bin $Dist_i$ as shown in Figure 3. The centroid of this area, which is the center value of the output bin since the bin is symmetrical, is then taken and multiplied with $\mu_i(rss_1)$, and all such sums are added. This sum is then divided by the sum of $\mu_i(rss_1)$ to yield distance d_1:

$$d_1 = \left(\frac{\sum_{i \in Rules} \mu_i(rss) * c(Dist_i)}{\sum_{i \in Rules} \mu_i(rss)} \right) \tag{4.1}$$

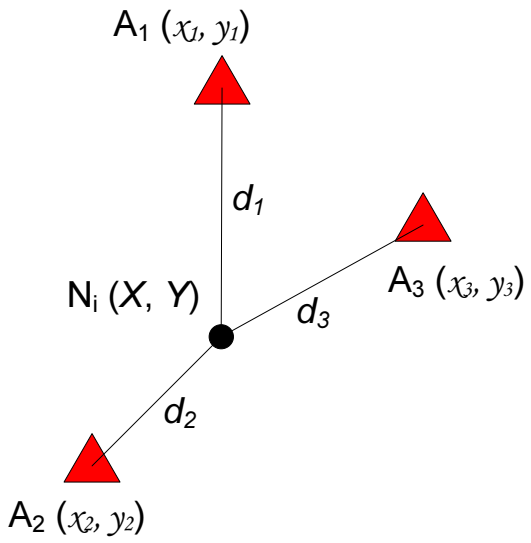

Fig. 2. A sensor node N_i with fuzzy coordinates X and Y, to be located using three anchors positioned at (x_1, y_1), (x_2, y_2) and (x_3, y_3)

This method is called the center average defuzzification. The output will be a crisp number d_1 which is again fuzzified into

$$D_1 = Fz(d_1)$$

where Fz denotes fuzzification. Repeating this process for each anchor, we end up with D_1, D_2 and D_3.

Now, let the coordinates of the anchors be (x_i, y_i). We have,

$$\begin{aligned} F_1 &= (X - x_1)^2 + (Y - y_1)^2 - D_1^2 = 0 \\ F_2 &= (X - x_2)^2 + (Y - y_2)^2 - D_2^2 = 0 \\ F_3 &= (X - x_3)^2 + (Y - y_3)^2 - D_3^2 = 0 \end{aligned} \tag{4.2}$$

Here, X and Y are fuzzy numbers which represent the location of the node. Note that D_i is a fuzzy number also. In order to solve this non-linear system of equations in two fuzzy variables, we employ the fuzzy variant of the iterative classical Newton

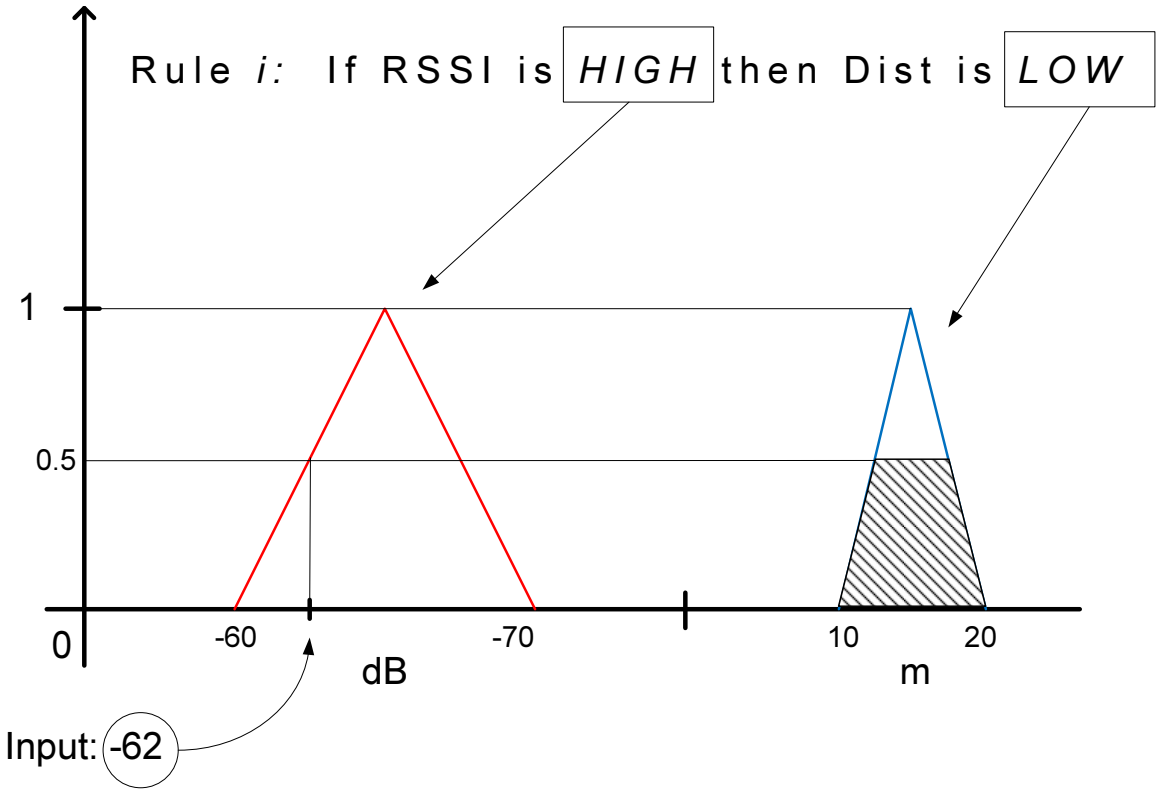

Fig. 3. Applying an input RSSI of -62dB to Rule i

method based on the Jacobian matrix. To use this method, fuzzy numbers need to be represented in their parametric form:

$$X = (\underline{X}, \overline{X})$$

where $\underline{X}$ is a continuous bounded non-decreasing function (effectively the "left half" of the membership function). Likewise for $\overline{X}$. Now consider the special case of the triangular membership function, as depicted in Figure 1 and defined by (a, b, c) where the membership is 0 at a, increases linearly to 1 at b and again decreases to 0 at c.

The parametric representation of X, in terms of a variable $r \in [0, 1]$ is:

$$X = (a + (b - a)r, c - (c - b)r)$$

Looking back to the system of equations, we seek to represent it in the parametric form. Without any loss of generality, we can assume that X and Y are positive. The system can now be split into:

$$\begin{aligned} \underline{F_1} &= (\underline{X} - x_1)^2 + (\underline{Y} - y_1)^2 - \underline{D_1}^2 = 0 \\ \underline{F_2} &= (\underline{X} - x_2)^2 + (\underline{Y} - y_2)^2 - \underline{D_2}^2 = 0 \\ \underline{F_3} &= (\underline{X} - x_3)^2 + (\underline{Y} - y_3)^2 - \underline{D_3}^2 = 0 \end{aligned} \tag{4.3}$$

and

$$\begin{aligned} \overline{F_1} &= (\overline{X} - x_1)^2 + (\overline{Y} - y_1)^2 - \overline{D_1}^2 = 0 \\ \overline{F_2} &= (\overline{X} - x_2)^2 + (\overline{Y} - y_2)^2 - \overline{D_2}^2 = 0 \\ \overline{F_3} &= (\overline{X} - x_3)^2 + (\overline{Y} - y_3)^2 - \overline{D_3}^2 = 0 \end{aligned} \tag{4.4}$$

We can now construct the Jacobian J as:

$$J = \begin{bmatrix} \underline{F_1}_{\underline{X}} & \underline{F_1}_{\overline{X}} & \underline{F_1}_{\underline{Y}} & \underline{F_1}_{\overline{Y}} \\ \overline{F_1}_{\underline{X}} & \overline{F_1}_{\overline{X}} & \overline{F_1}_{\underline{Y}} & \overline{F_1}_{\overline{Y}} \\ \underline{F_2}_{\underline{X}} & \underline{F_2}_{\overline{X}} & \underline{F_2}_{\underline{Y}} & \underline{F_2}_{\overline{Y}} \\ \overline{F_2}_{\underline{X}} & \overline{F_2}_{\overline{X}} & \overline{F_2}_{\underline{Y}} & \overline{F_2}_{\overline{Y}} \\ \underline{F_3}_{\underline{X}} & \underline{F_3}_{\overline{X}} & \underline{F_3}_{\underline{Y}} & \underline{F_3}_{\overline{Y}} \\ \overline{F_3}_{\underline{X}} & \overline{F_3}_{\overline{X}} & \overline{F_3}_{\underline{Y}} & \overline{F_3}_{\overline{Y}} \end{bmatrix} \tag{4.5}$$

$$J = \begin{bmatrix} 2(\underline{X} - x_1) & 0 & 2(\underline{Y} - y_1) & 0 \\ 0 & 2(\overline{X} - x_1) & 0 & 2(\overline{Y} - y_1) \\ 2(\underline{X} - x_2) & 0 & 2(\underline{Y} - y_2) & 0 \\ 0 & 2(\overline{X} - x_2) & 0 & 2(\overline{Y} - y_2) \\ 2(\underline{X} - x_3) & 0 & 2(\underline{Y} - y_3) & 0 \\ 0 & 2(\overline{X} - x_3) & 0 & 2(\overline{Y} - y_3) \end{bmatrix} \tag{4.6}$$

Initial guesses of X and Y can be updated as follows: for every iteration compute a matrix Δ:

$$\Delta = \begin{bmatrix} \underline{h}(r) \\ \overline{h}(r) \\ \underline{k}(r) \\ \overline{k}(r) \end{bmatrix} \tag{4.7}$$

where h_1 etc are defined as the incremental updates to the initial guess.

$$\begin{aligned} \underline{X}(r) &= \underline{X}(r) + \underline{h}(r) \\ \overline{X}(r) &= \overline{X}(r) + \overline{h}(r) \\ \underline{Y}(r) &= \underline{Y}(r) + \underline{k}(r) \\ \overline{Y}(r) &= \overline{Y}(r) + \overline{k}(r) \end{aligned} \tag{4.8}$$

The set of equations evaluated at the initial guess is:

$$F = \begin{bmatrix} \underline{F_1} \\ \overline{F_1} \\ \underline{F_2} \\ \overline{F_2} \\ \underline{F_3} \\ \overline{F_3} \end{bmatrix} \tag{4.9}$$

The equation that connects them is:

$$\Delta = -J^{-1}F$$

First, the initial guess (X_0, Y_0) is computed from the average of the coordinates of the anchors. Then, J and F are computed for this initial guess. The incremental update Δ is then calculated and applied to X and Y. J and F are then computed for the new values and the process is repeated till Δ is sufficiently close to zero.

4.2 Fuzzy Grid Prediction

Consider the area in which the network is deployed to be subdivided into grids. Assume that a 500×500 area is divided into square grids of side 100, which leaves a matrix of 5 rows and 5 columns. Assume again that there is an anchor at the center of every grid. We now propose to tackle the problem of not having enough anchors to perform multi-lateration.

The key idea is very simple - if the distance to an anchor is less, it is highly probable that the node is in the same grid which the anchor is at the center of.

The inspiration for this part of the algorithm is [23]. We aim to construct a fuzzy inference system which will help predict a grid in the deployment area which has a high probability of containing the node in question, as shown in Figure 4. The

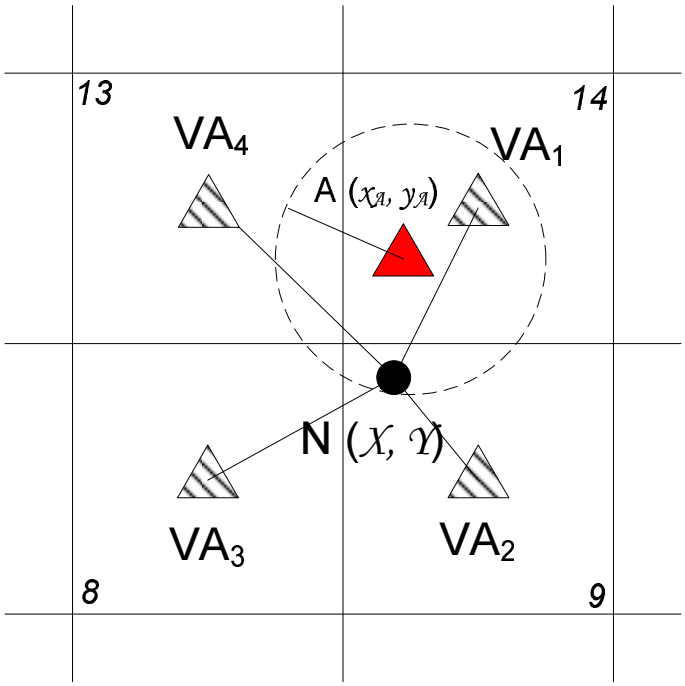

Fig. 4. Grid prediction setup - only 4 of 25 grids are shown

center of gravity of the lamina defined by the intersection of he square defined by this grid and the radio range defined by the anchor will yield the required location. The system consists of fuzzy rules as before. The data which serves as input is the calculated average distance $\overline{D}$ to each virtual anchor. A sample rule is:

IF (Dist$_{grd_0}$ is D_0) and ... and (Dist$_{grd_n}$ is D_n) **THEN** Prob$_{g_{[0-n]}}$ is $P_{[0-n]}$

where Dist$_{grd_0}$ is the calculated average distance to the virtual anchor situated at the center of grid 0 and $P_{g_{[0-n]}}$ is probability that the node is in grids 0 to n etc.

We first train the system. This is done by considering the actual position of the node (which is not available except in training) and then generating probabilities.

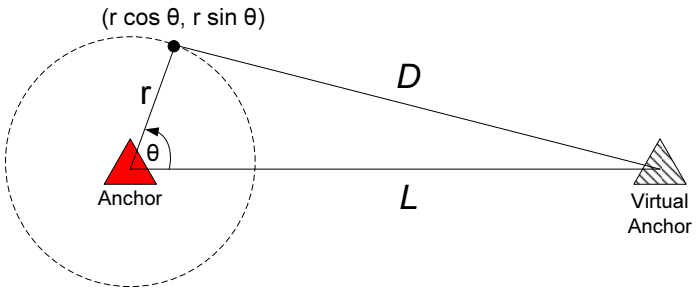

Fig. 5. Average distance between anchor and VA

4.2.1 Virtual Anchors

As we see in [23], the fuzzy prediction performs best when there is an anchor at the center of every grid. This is unreasonable to expect in a highly mobile network. Therefore, we propose to replace each anchor with mathematically equivalent anchors situated at the center of every grid.

Consider a node and an anchor which is its neighbor. Take all possible virtual anchors and discard the ones which are at a distance of more than $2R$ from the anchor. This forms the set of possible virtual anchors. Then, calculate the average distance from the node to each of the virtual anchors. This scenario, illustrated in Figure 5, can be mathematically calculated as follows: the average distance from the virtual anchor to all the points on the circumference of a hypothetical circle at whose center the real anchor is located. The radius of the circle is the calculated distance between the node and the anchor which is obtained from the fuzzy inference engine contained in FNLS (no non-linear equations are constructed or solved). Since the node can be anywhere on the circumference of this circle, the average distance $\overline{D}$ can be calculated using the following equation:

$$\begin{aligned}\overline{D} &= \frac{1}{2\pi}\int_0^{2\pi}\sqrt{(L\ -\ r\ \cos\theta)^2+(r\ \sin\theta)^2}\ d\theta \\ &= \frac{1}{2\pi}\int_0^{2\pi}\sqrt{L^2+r^2-2Lr\cos\theta}\ d\theta \\ &= \frac{(L-r)}{\pi}E\big[\pi|\frac{-4Lr}{(L-r)^2}\big]\end{aligned} \tag{4.10}$$

where $E[x|m]$ is the incomplete elliptic integral of the second kind.

This distance is then calculated for each of the possible virtual anchors, which is nothing but the input to the Fuzzy inference system. The output will be a grid. The center of gravity of the lamina defined by the intersection is then calculated. Thus, a location is obtained.

4.3 Example of Fuzzy Logic Based Localization

Consider 3 anchors at the sides of a triangle - $(0,0)$ $(10,0)$ and $(5,15)$. Let the node to be localized be at the centroid which is $(5,5)$. The distances the node would then calculate is $5\sqrt{2} = 7.0711$. Let us assume the bin corresponding to this to be $(6,7,8)$. We now have the system of equations as:

$$F_1 = (X-0)^2 + (Y-0)^2 - (6,7,8)^2 = 0 \tag{4.11}$$

$$F_2 = (X-10)^2 + (Y-0)^2 - (6,7,8)^2 = 0 \tag{4.12}$$

$$F_3 = (X-5)^2 + (Y-15)^2 - (6,7,8)^2 = 0 \tag{4.13}$$

Assume X and Y to be $(5,6,7)$ - it really is $(4,5,6)$. The parametric form would then be $(5+r, 7-r)$ as explained before. Also,

$$\underline{F_1} = (\underline{X} - 0)^2 + (\underline{Y} - 0)^2 - \underline{(6,7,8)}^2$$
$$\overline{F_1} = (\overline{X} - 0)^2 + (\overline{Y} - 15)^2 - \overline{(6,7,8)}^2$$

which can then be simplified to

$$\underline{F_1} = (5 + r - 0)^2 + (5 + r - 0)^2 - (6 + r)^2$$
$$\overline{F_1} = (7 - r - 0)^2 + (7 - r - 15)^2 - (8 - r)^2$$

Similarly the Jacobian can now be constructed as (first 2 rows only):

$$J = \begin{bmatrix} 2(5+r-0) & 0 & 2(5+r-0) & 0 \\ 0 & 2(7-r-0) & 0 & 2(7-r-0) \end{bmatrix} \tag{4.14}$$

The pseudo-inverse of a matrix with symbolic elements is computationally expensive, especially for motes. Instead of inverting J which contains a symbolic element r, we can instead compute two non-symbolic inverses (for $r = 0$ and $r = 1$), and then combine the results. The "cost" for this "free lunch" is that the solution will be a perfect triangular fuzzy number, and not a fuzzy number with little variation. However, the accuracy lost with this method is extremely small.

First, we make the simple substitution $r = 0$ in J and F, to yield

$$\Delta_0 = J_0^{-1} F_0$$

Now if the solution (X, Y) is expressed in simple form (not in parametric form) as (x_A, x_B, x_C) and (y_A, y_B, y_C), we have:

$$\Delta_0 = \begin{bmatrix} \delta x_A \\ \delta x_C \\ \delta y_A \\ \delta y_C \end{bmatrix} \tag{4.15}$$

where δx_A is the incremental update to x_A. This is obvious since the left half of any fuzzy number in parametric form $(a + (b - a)r)$ computes to a when $r = 0$. The same argument holds for the right half also.

Subsequently, substituting $r = 1$ yields Δ_1:

$$\Delta_1 = \begin{bmatrix} \delta x_B \\ \delta x_B \\ \delta y_B \\ \delta y_B \end{bmatrix} \tag{4.16}$$

After this step, we now have the new (x_A, x_B, x_C) and (y_A, y_B, y_C), which is the input for the next iteration. The process is repeated until sufficient accuracy is obtained.

CHAPTER V

LOCALIZATION SYSTEM DESIGN

The system design for the proposed node localization scheme is depicted in Figure 6. The system components that implement the proposed Fuzzy Multilateration and Fuzzy Grid Prediction techniques described in Chapter IV are the Fuzzy Non Linear System (FNLS) and Fuzzy Grid Prediction System (FGPS). Both FNLS and FGPS consist of a Fuzzy Inference System (FIS) which basically applies the input to the rule database. For FNLS, the FIS handles RSSI and distance; for FGPS it handles distance and probability. These distances are used in the two constituent components - the Fuzzy Non Linear System (FNLS) and the Fuzzy Grid Prediction System (FGPS). FNLS, after inferring a distance, builds a fuzzy nonlinear system of equations in order to compute the location of the node as a fuzzy number. FGPS uses the defuzzified distance from FNLS to first calculate the distances to virtual anchors, and then uses the distances as input to the FIS. A vector of probabilities is returned as the output.

Once all the output is sent to the node, it decides whether to solve a NLS and infer a location or use the FGPS results based on the number of anchors it hears. If in case it has only anchor among its neighbors, it chooses the grid with the maximum probability as its location. In order to further narrow the region, it geometrically computes the area around the anchor and then uses it to narrow down the grid to a smaller area. This is because the node HAS to be within hearing distance from the anchor.

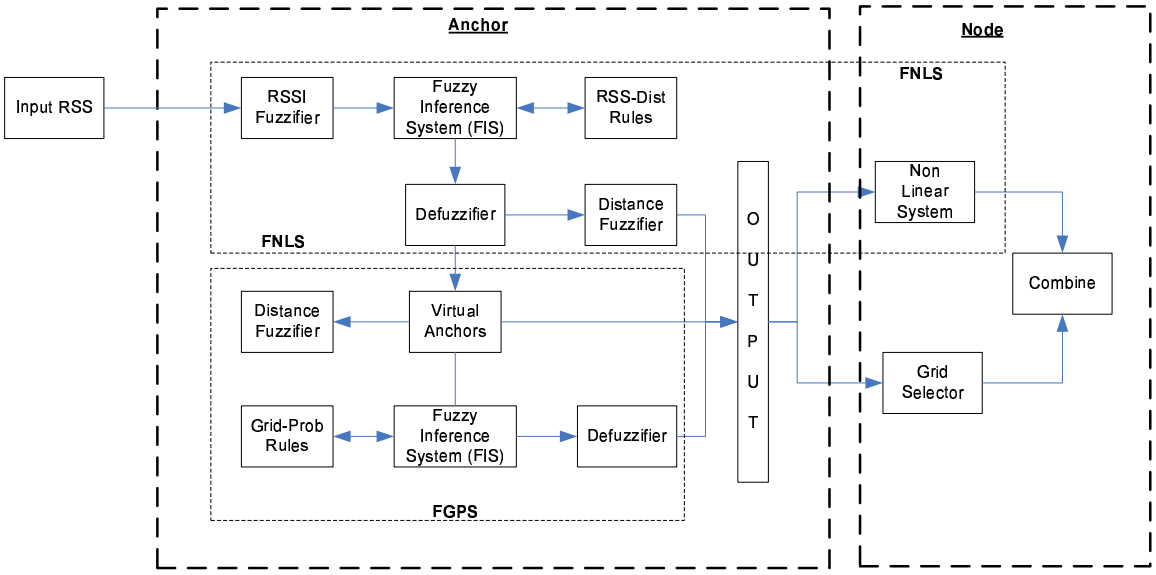

Fig. 6. Overall system decision process

5.1 A Distributed Protocol

We can implement the above system for localization as a protocol for mobile nodes via two kinds of messages - a HELLO message which anchors use to broadcast their location and build rules, and a HELP message which node use to notify anchors that they need to localize. The anchors are assumed to have a little more computing power than ordinary nodes. They can then maintain the fuzzy rules needed for FGPS and FNLS. Once the nodes are deployed, the anchors broadcast a HELLO message advertising their location (Algorithm 5.1, step 1) and their virtual anchors. Whenever another anchor hears a HELLO, it uses its own location and the RSS of the incoming message to train the FIS of the FNLS system (step 7). Then, the FIS of the FGPS system is trained using the virtual anchors of the sending anchor (step 11). Thus, one rule each for the FNLS and FGPS systems are built. These rules are used whenever a HELP message, which is essentially a request for localization is sent out by a node.

The pseudocode for the localization protocol is shown in Algorithm 5.1 and Algorithm 2.

When an anchor hears a HELP, it first calculates the RSS of the message via internal methods (Algorithm 5.1, step 14). It uses this as input to the FIS of the FNLS system, which essentially defuzzifies the RSS into a distance using the rule database previously built (step 15). Using this distance, it calculates the distances to its virtual anchors (step 17). These set of distances are once again the input to the FGPS system (step 18). A vector of probabilities is returned, which in turn is sent back to the sending node along with the other information (step 19).

A node sends out a HELP message whenever it desires to localize itself (Algorithm 2, step 1). Anchor(s) reply to this message and the number of such replies is obviously the number of anchors in the node's vicinity (step 3). If this number happens to be unity, the node chooses to construct two geometrical objects - one, the grid which corresponds to the maximum probability in the probability vector; two, a circle with the anchor at the center and with a radius equal to the distance between the anchor and the node. The intersection of these objects will be the location of the node (steps 8-12).

If it hears two or more than two anchors, the node chooses to construct a FNLS and then solve it iteratively (steps 14-16). The solution of this system eventually returns a location. In case the node hears no anchors at all, the most recently calculated location is claimed as the current location.

Algorithm 1 FuzLoc Protocol - Anchors

1: $[VA] \leftarrow$ *FGPS.getVirtualAnchors* ▷ VA of self

2: *BroadcastHello(VA)*

3: **procedure** RecvHello(anchor n)

4: $rss \leftarrow$ *Radio.getRSS()*

5: $loc \leftarrow$ *Message.parseLocation()* ▷ Loc of sender

6: $dist \leftarrow$ Distance to sender

7: *FNLS.train(rss, dist)*

8: $[VA] \leftarrow$ *Message.parseVA()* ▷ VA of sender

9: $[dist] \leftarrow$ Calculate distances to virtual anchors

10: $[prob] \leftarrow$ Calculate probabilities

11: *FGPS.train(dist, prob)*

12: **end procedure**

13: **procedure** RecvHelp(anchor n)

14: $rss \leftarrow$ *Radio.getRSS()*

15: $dist \leftarrow$ *FNLS.getDist(rss)*

16: $[VA] \leftarrow$ *FGPS.getVirtualAnchors*

17: $[VA.dist] \leftarrow$ *FPGS.getDists(VA)*

18: $[VA.prob] \leftarrow$ *FPGS.defuzzify(VA.dist)*

19: *Radio.reply(dist, VA.dist, VA.prob)*

20: **end procedure**

Algorithm 2 FuzLoc Protocol - Nodes

```
1: BroadcastHelp()                                          ▷ Initiates localization
2: [info] ← ConsolidateHelpReplies
3: anchors ← Count(info)
4: procedure LOCALIZE(node n)
5:     if anchors = 0 then
6:         loc ← Previous Location
7:     else if anchors = 1 then
8:         [prob] ← info[0].parseVAProb()
9:         grid ← Max(prob).index
10:        dist, center ← info[0].parseAnchorLoc()
11:        circle ← ConstructCircle(dist, center)
12:        loc ← SolveIntersection(grid,circle)
13:    else
14:        [dists] ← info.parseDistances()
15:        [centers] ← info.parseLocations()
16:        loc ← solveFNLS(dists,centers)
17:    end if
18: end procedure
```

5.2 Training

5.2.1 FNLS - Training

Training this system involves fuzzifying both the RSS and the distance during the training period. First, anchors exchange locations during the HELLO broadcast period. The RSS rss of the incoming message is calculated. The membership rss in each RSS bin is calculated. The bin which corresponds to the maximum membership is chosen as the corresponding bin $rbin$. Similarly, the distance between anchors is computed since both the anchors know each others locations. The distance d between anchors is similarly fuzzified into a distance bin $dbin$. The following rule will then be inserted into the ruleset:

IF RSS is $rbin$ **THEN** distance is $dbin$

5.2.2 FGPS - Training

Training this system involves fuzzifying the RSS from each valid virtual anchor and the calculated probabilities. When an anchor receives a HELLO, it also receives the list of valid virtual anchors of the sending anchor. It then calculates distances to each of those virtual anchors using the `EllipticE` method. As for the calculation probabilities, the node first locates in own grid.

CHAPTER VI

PERFORMANCE EVALUATION

Performance evaluation was carried out using a multi-method simulator written by the authors of [6]. Along with MCL, Centroid and Amorphous Localization are implemented in the simulator. There is also a "Perfect Fuzzy" algorithm, which is essentially FuzLoc but with a 100% accurate fuzzy inference subsystem, which is impossible to achieve. This ghost algorithm quantifies the errors caused by the topology of the network (e.g., number of anchors), and not due to the learned intelligence gathered by the anchors. We noticed that a significant portion of the error is due to the absence of anchors at an anchor density of 10%.

The simulation setup is structured as follows. For each value of each metric (i.e., each data point), there are 10 iterations. Each iteration consists of each node taking 50 "steps". The mobility model chosen is random waypoint. Nodes move towards a fixed destination (which is not visible to the nodes) in steps. At each step, it moves a maximum distance denoted by max_v. The localization error is averaged for each method of localization over 50 steps, and then over 10 iterations. Since FuzLoc returns a fuzzy location, it is first defuzzified into a crisp location by considering the center values b of each of the fuzzy numbers representing the abscissa and the ordinate. The default parameters used are shown in Table I.

6.1 Degree of Irregularity

The degree of irregularity helps quantify the real non-spherical radio range experienced by antennae. Figure 7 illustrates the radio pattern caused by DoIs of 0.0 (ideal

Table I. Default simulation parameters

Grid	500×500
Nodes	320
Seeds	32
DoI	0.4
Radio Range	50
Iterations	10
Fuzzy Bins	10
Bin Type	Triangular
Defuzzification	Center-average

case) and 0.4. As we can see, the change in range is at most 40% for a DoI of 0.4.

Simulation was carried out with varying DoI. All other parameters were kept constant. Figure 8 shows the runaway behavior of MCL. This effect of errors being compounded due to polluted samples has been investigated as the "kidnapped robot problem" in the world of robot localization. The kidnapped robot test verifies whether the localization algorithm is able to recover from localization failures, as signified by the sudden change in location due to "kidnapping". It is well known that MCL based algorithms suffer from this problem; many remedies have also been studied and implemented. It has been shown that such uncorrected algorithms collapse when the

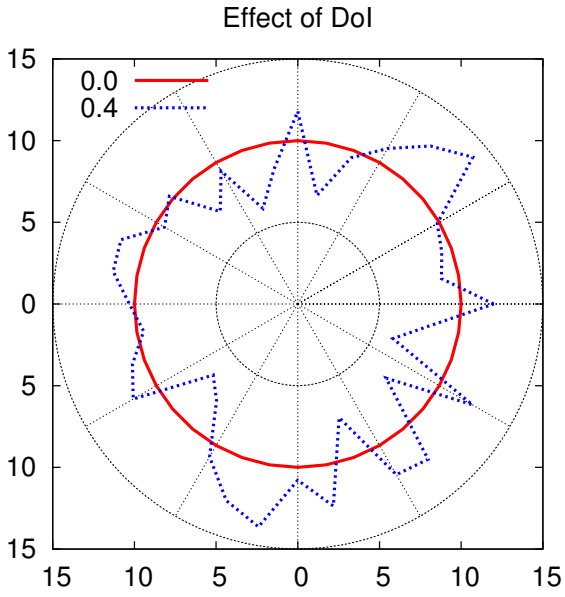

Fig. 7. Illustration of radio pattern for different DoI

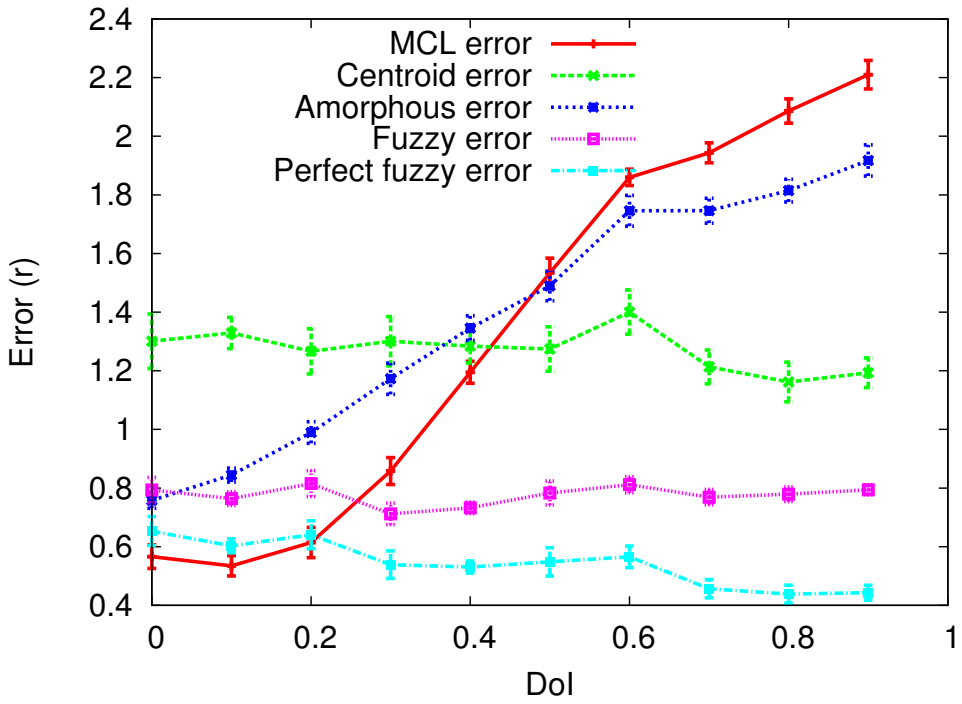

Fig. 8. Effect of DoI

observed sample is quite far from the estimated sample.

6.2 Anchor Density

The anchor density is a critical parameter for anchor-based localization schemes. In Figure 9, we see the impact of anchor density on the localization schemes. The number of anchors vary from 10% (32 anchors) to 50% (160 anchors). The simulation was done for a DoI of 0.4. We see that the accuracy of MCL suffers since the samples become increasingly polluted because as the number of anchors increases, the amount of "misinformation" also increases due to high DoI. This mismatch of observed and calculated values causes the error to compound. Centroid performs better with increasing anchor density. The error for Amorphous does not improve much, as noted in [6].

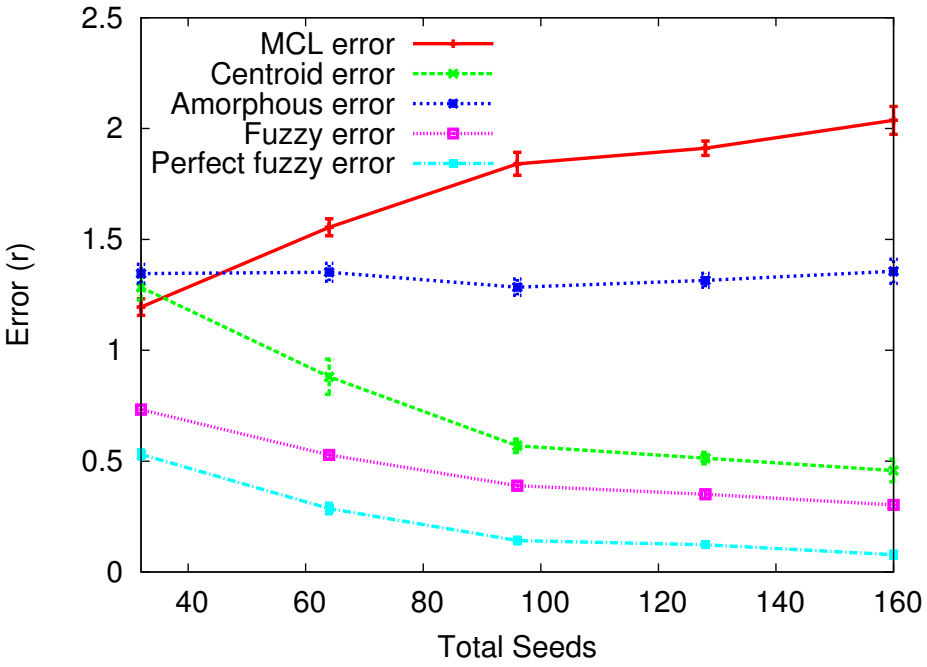

Fig. 9. Effect of varying anchor density

The same evaluation of the effect of anchor density has been done for a DoI of 0.2, as shown in Figure 10.

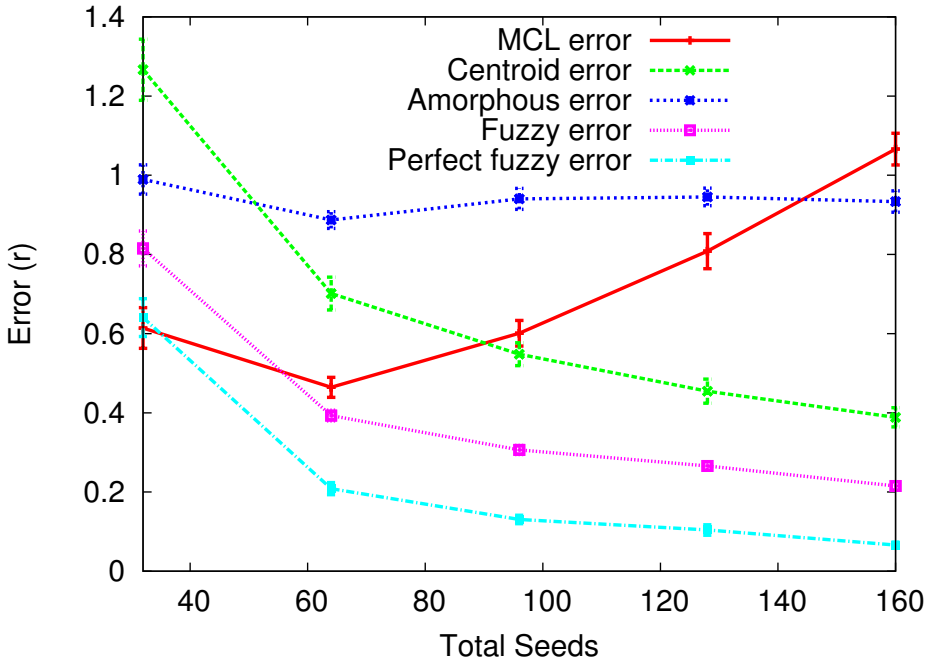

Fig. 10. Effect of anchor density at a DoI of 0.2

6.3 Node Density

As shown in Figure 11 Node density does not significantly affect the algorithms since the number of anchors is kept constant throughout at 10%. Amorphous, however, needs the propagated messages to be delivered to the node. Hence, a higher node density means more neighbors, which means better communication between nodes.

The same node density test was run at a lower DoI with the results as shown in Figure 12. It is to be noted that a higher DoI means that a node may not always have the same neighbors in adjacent steps of the simulation, since the radio range of a node differs at every instant.

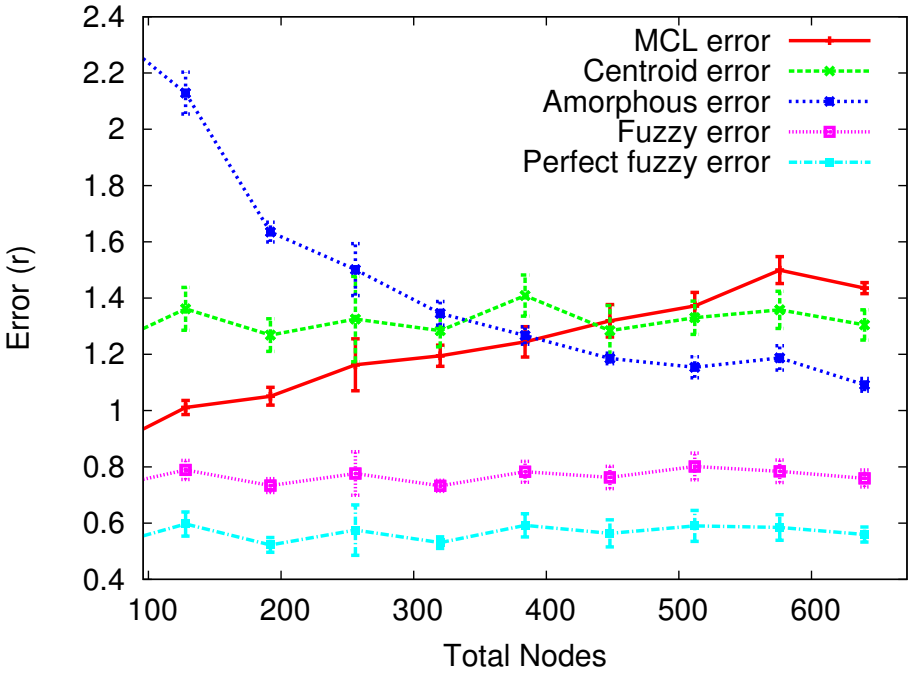

Fig. 11. Effect of node density

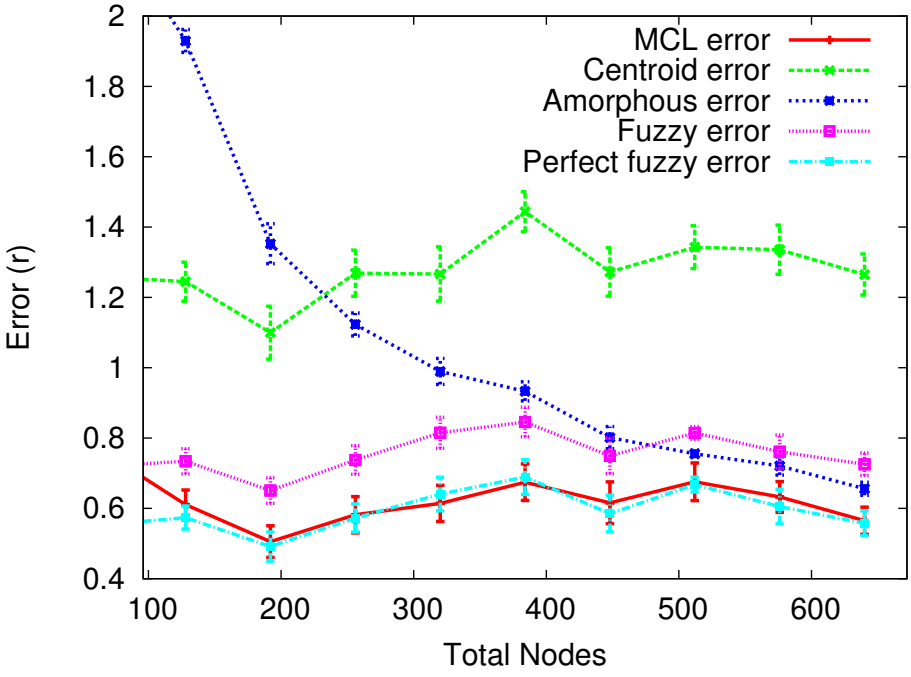

Fig. 12. Effect of node density at a DoI of 0.2

6.4 Number of Bins

The number of bins in the fuzzy system is a design parameter - the greater the number of bins, better will be the accuracy of the system. As the number of bins increases, more and more RSSs will find a bin with high membership. This causes more variety in rules and more rules to fire strongly during the fuzzy inference process. Hence, a better output will be obtained. Changing this number should not affect the other algorithms. As we see in Figure 13, the other methods remain invariant whereas FuzLoc experiences decreasing error with an increase in the number of bins.

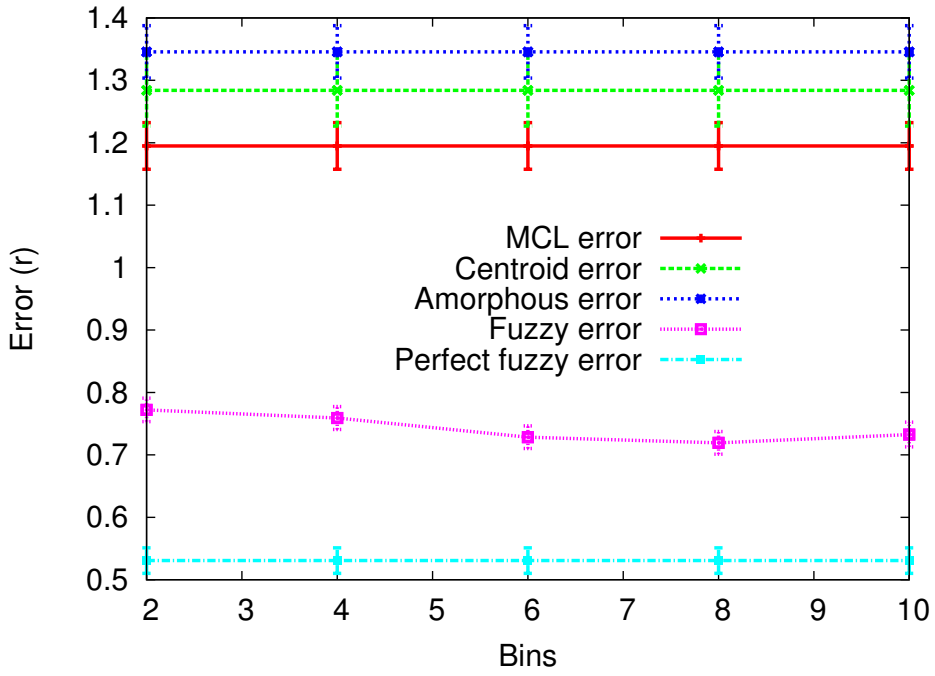

Fig. 13. Effect of number of fuzzy bins

6.5 Maximum Velocity

MCL assumes that nodes know the maximum velocity, whereas none of the other algorithms do. This information is then used to filter the samples, which results in accurate localization. FuzLoc suffers from low anchor density at higher node speeds. Hence the localization errors increases, but does not increase by a large percentage. The results are depicted in Figure 14.

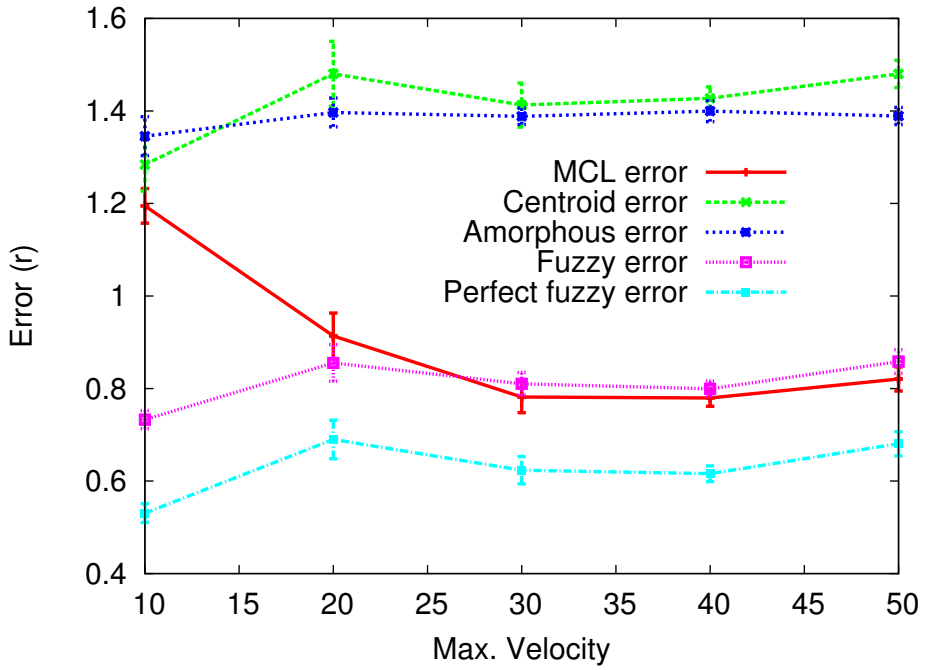

Fig. 14. Effect of maximum velocity

6.6 Fuzzy Performance

The following figure shows the performance of the FNLS FIS engine, which is the main subsystem. On the X axis is the input distance and on the Y is the defuzzified output distance. After training the system using random RSS-Distance pairs, RSS

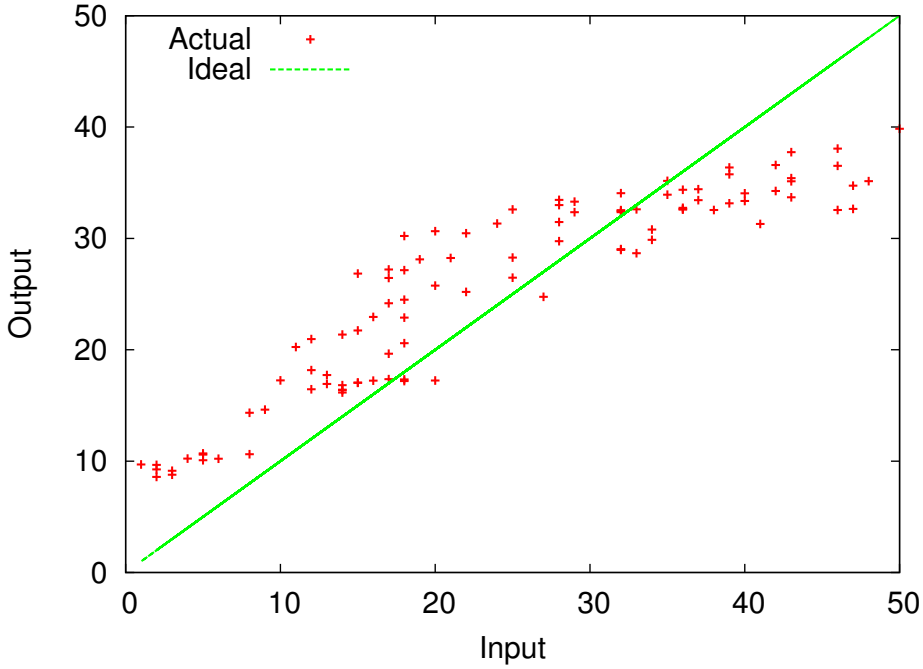

Fig. 15. Performance of the FNLS FIS subsystem

values deduced from distances were fed into the system, so that a distance should be inferred. Ideally, it should be a straight line with a slope of unity, however, the actual output is plotted in the Figure 15. The data for the figure was gathered after the fuzzy system was trained with 30 random pairs of data.

6.7 Memory Overhead

A typical FIS does not require much storage capacity. If there were 8 bins, for e.g., a single byte can represent a bin. Hence, each FNLS rule requires just 2 bytes of storage. Typically, an anchor creates around 30 rules during the period of deployment which translates to 60 bytes of storage. The FGPS FIS however, requires 50 bytes for each rule (25 bins in the input, 25 in the output). Note that regular nodes do not store rules, only the anchors which are a fraction of the total number store rules. Moreover,

due to the nature of the triangular bin shapes, simple calculations are required in order to fuzzify and defuzzify. The only caveat is the inversion of matrices that is required.

MCL requires at least 50 samples for low localization error. Each sample requires a weight. Centroid does not store any history and thus has the least storage requirement. Amorphous requires storing the announcements made by the seeds which are flooded throughout the network. If there are 320 nodes, 32 of which are anchors, MCL requires each node to store 50 samples. Each sample has an abscissa and an ordinate, each of at least 4 bytes. Hence, MCL requires around $(50 \times 4 \times 2 \times 320)$ = 128000 bytes. Fuzzy on the other hand requires around 1500 bytes for FGPS and around 60 for FNLS = (1560×32) = 49920 bytes which is roughly 40% of the storage MCL requires.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

Localization in indoor environments suffers from errors due to peculiarities of the environment. Our method overcomes these errors even in the presence of mobility. Instead of attempting to model the radio wave propagation so that an accurate distance can be inferred from RSS, we instead use fuzzy logic to treating the quantities involved using a different semantic. This way, the imprecision in the measured RSS was encoded into a fuzzy variable. Subsequently, the RSS was converted into a location in the fuzzy domain itself using a nonlinear system of fuzzy variables. Our method is anchor based, meaning that error reduces with increasing percentage of anchors. Finally, our method does not require much storage and processing as compared to MCL.

On a general note, fuzzy logic can be applied to WSNs in a variety of ways. The basic premise of using FL is that the system does not need to be modeled - only a few input-output pairs are required for the FL system to learn and behave as if it were the real system. This makes it ideal for WSNs since it is deployed in a variety of environments. For example, localization using ultrasonic frequencies also suffers from multipath and such related problems. The time of arrival can be related to the time of emission of the wave using a fuzzy rule. Basically, any two related quantities can be related in a fuzzy inference system. We intend to test FuzLoc using mobile robots which can communicate using RF.

Since FL has been shown to be quite reliable when applied to control theory, it can be applied in a distributed way to WSNs. Applications of distributed feedback

control include beamforming for sensor networks and power control. Again, there are a variety of applications for control that can use a reliable fuzzy inference system as a backend.

REFERENCES

[1] D. Niculescu and B. Nath, "Dv based positioning in ad hoc networks," *Telecommunication Systems*, vol. 22, no. 1, pp. 267–280, January 2003.

[2] C. Wang and L. Xiao, "Sensor localization in concave environments," *ACM Trans. Sen. Netw.*, vol. 4, no. 1, pp. 1–31, 2008.

[3] R. Nagpal, H. E. Shrobe, and J. Bachrach, "Organizing a global coordinate system from local information on an ad hoc sensor network," in *IPSN*, 2003, pp. 333–348.

[4] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low cost outdoor localization for very small devices," *IEEE Personal Communications Magazine*, vol. 7, no. 5, pp. 28–34, October 2000.

[5] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *MobiCom '03: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, 2003, pp. 81–95.

[6] L. Hu and D. Evans, "Localization for mobile sensor networks," in *MobiCom '04: Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, 2004, pp. 45–57.

[7] M. Rudafshani and S. Datta, "Localization in wireless sensor networks," in *IPSN '07: Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, 2007, pp. 51–60.

[8] L. Klingbeil and T. Wark, "A wireless sensor network for real-time indoor localisation and motion monitoring," in *IPSN '08: Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, 2008, pp. 39–50.

[9] M. Maróti, P. Völgyesi, S. Dóra, B. Kusý, A. Nádas, Ákos Lédeczi, G. Balogh, and K. Molnár, "Radio interferometric geolocation," in *SenSys '05: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, 2005, pp. 1–12.

[10] H. lin Chang, J. ben Tian, T.-T. Lai, H.-H. Chu, and P. Huang, "Spinning beacons for precise indoor localization," in *SenSys '08: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, 2008, pp. 127–140.

[11] P. Bahl and V. N. Padmanabhan, "Radar: an in-building rf-based user location and tracking system," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, vol. 2, 2000, pp. 775–784 vol.2.

[12] D. Niculescu and B. Nath, "Ad hoc positioning system (aps) using aoa," in *INFOCOM 2003: Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, April 2003, pp. 1734–1743 vol.3.

[13] K. Whitehouse and D. Culler, "Calibration as parameter estimation in sensor networks," in *WSNA '02: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, 2002, pp. 59–67.

[14] N. Patwari and I. Alfred O. Hero, "Using proximity and quantized rss for sensor localization in wireless networks," in *WSNA '03: Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks And Applications*, 2003, pp. 20–29.

[15] X. Shen, J. W. Mark, and J. Ye, "Mobile location estimation in cdma cellular networks by using fuzzy logic," *Wirel. Pers. Commun.*, vol. 22, no. 1, pp. 57–70, 2002.

[16] R. Stoleru, P. Vicaire, T. He, and J. A. Stankovic, "Stardust: a flexible architecture for passive localization in wireless sensor networks," in *SenSys '06: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, 2006, pp. 57–70.

[17] R. Stoleru, T. He, J. A. Stankovic, and D. Luebke, "A high-accuracy, low-cost localization system for wireless sensor networks," in *SenSys '05: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, 2005, pp. 13–26.

[18] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, "Localization from mere connectivity," in *MobiHoc '03: Proceedings of the 4th ACM International Symposium on Mobile ad hoc Networking & Computing*, 2003, pp. 201–212.

[19] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *MobiCom '01: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, 2001, pp. 166–179.

[20] C. Savarese, J. M. Rabaey, and K. Langendoen, "Robust positioning algorithms for distributed ad-hoc wireless sensor networks," in *USENIX Annual Technical Conference, General Track*, 2002, pp. 317–327.

[21] A. Baggio and K. Langendoen, "Monte Carlo localization for mobile wireless sensor networks," *Ad Hoc Netw.*, vol. 6, no. 5, pp. 718–733, 2008.

[22] M. Martins, H. Chen, and K. Sezaki, "Otmcl: Orientation tracking-based monte carlo localization for mobile sensor networks," in *Proceedings of the Sixth International Conference on Networked Sensing Systems (INSS 2009)*, Carnegie Mellon University, Pittsburgh, PA, Jun. 2009.

[23] X. Shen, J. W. Mark, and J. Ye, "User mobility profile prediction: an adaptive fuzzy inference approach," *Wirel. Netw.*, vol. 6, no. 5, pp. 363–374, 2000.

VITA

Harshavardhan Chenji Jayanth was born in Bangalore, India. After completing schooling and college at The Home School, Vijaya High School and the National College Jayanagar, he attended university at the National Institute of Technology Karnataka, Surathkal, India from 2003 to 2007. He received the degree of Bachelor of Technology in May, 2007. He entered graduate school at Texas A&M University, College Station in August, 2007 and graduated with his Master of Science in 2009. He can be reached at:

Permanent Address:
#22, 1^{st} B Cross, 33^{rd} Main, 7^{th} Block,
Banagirinagar, BSK 3^{rd} Stage,
Bangalore, India 560 085

Email: h.chenji at gmail.com

The typist for this thesis was Harshavardhan Chenji Jayanth.